



#5/647-10 5755
T.H. (P) 755
11/17/00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: LAEMMLE, B., et al. : Atty Dkt. No.: 99 P 7475 US01
Serial No.: 09/506,640 : Examiner: To Be Assigned
Filed: February 18, 2000 : Group Art Unit: 2755
For: MONIKER METHOD, APPARATUS, : Date: August 24, 2000
SYSTEM AND ARTICLE OF
MANUFACTURE

Certificate of Mailing

I hereby certify that this correspondence, and all correspondence referred to herein, is being deposited with the U.S. Postal Service in an envelope containing sufficient postage as first class mail and addressed to Assistant Commissioner For Patents, Commissioner of Patents and Trademarks, Washington, D.C. 20231 on the date indicated below.

8/24/00
Dated

Georgeana Heiss
Georgeana Heiss

THE ASSISTANT COMMISSIONER
FOR PATENTS
WASHINGTON, DC 20231

LETTER TO THE OFFICIAL DRAFTSMAN

S I R:

Transmitted herewith are ten (10) sheets of Formal Drawings for the above-identified application.

Applicant's undersigned attorney may be reached in our Iselin, New Jersey office by telephone at (732) 321-3009. All correspondence should continue to be directed to our below-listed address.

Respectfully submitted,

I. Marc Asperas
I. Marc Asperas
Reg. No. 37,274

Dated: August 24, 2000

Siemens Corporation
Intellectual Property Department
186 Wood Avenue South
Iselin, NJ 08830
(732) 321-3009

RECEIVED
AUG 30 2000
TC 2700 MAIL ROOM

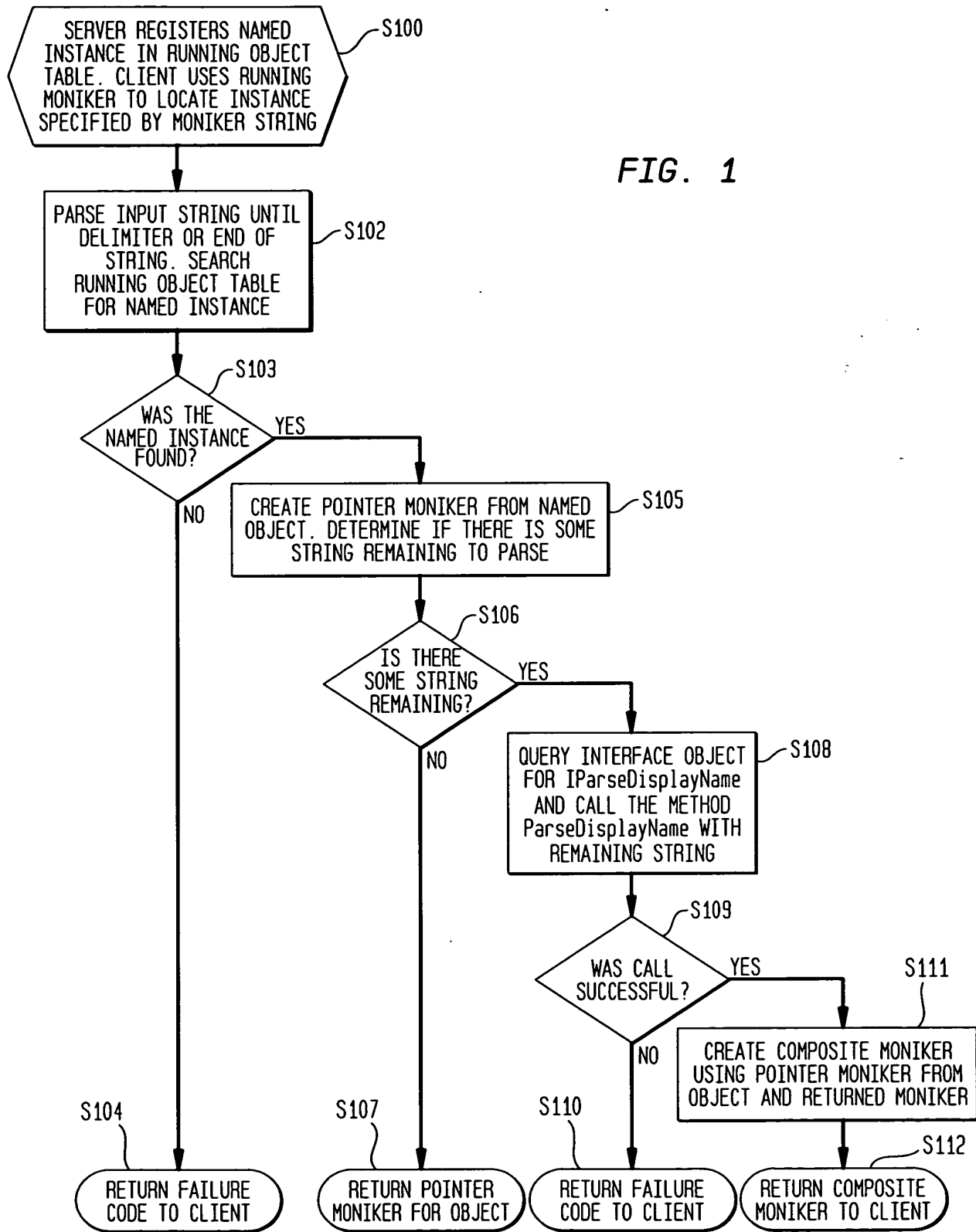


FIG. 2A

Running.idl

// Running.idl : IDL source for Running.dll

//

// This file will be processed by the MIDL tool to

// produce the type library (Running.tlb) and marshalling code.

import "oidl.idl";

import "ocidl.idl";

```

    [
        object,
        uuid(E7531917-289D-11D2-869F-080009DC2552),
        dual,
        helpstring ("IRunning Interface"),
        pointer_default (unique)
    ]
    interface IRunning : IDispatch
    [
        [id(1), helpstring("method RegisterInstanceName")] HRESULT RegisterInstanceName (BSTR bstr
        ItemName, IUnknown * pUnk, long * lCookie);
        [id(2), helpstring("method UnregisterInstanceName")] HRESULT UnregisterInstanceName(long
        lCookie);
    ];
    [
        uuid(E7531908-289D-11D2-869F-080009DC2552),
        version(1.0),
        helpstring ("Running 1.0 Type Library")
    ]
    library RUNIINGLib
    [
        importlib("stdole32.tib");
        importlib("stdole2.tib");

        [
            uuid(E7531918-289D-11D2-869F-080009DC2552),
            helpstring("Running Class")
        ]
        coclass Running
        [
            [default] interface IRunning;
            interface IparseDisplayName;
        ];
    ];

```

FIG. 2B

Running.rgs

HKCR

(

Running.1 = s 'Running Class'

(

CLSID = s '(E7531918-289D-11D2-869F-080009DC2552)'

)

Running = s "Running Class"

(

CLSID = s ;(E7531918-289D-11D2-869F-080009DC2552) "

)

NoRemove CLSID

(

ForceRemove (E7531918-289D-11D2-869F-080009DC2552) = s 'Running Class'

(

val AppID = s '(E7531918-289D-11D2-869F-080009DC2552)'

ProgID = s 'Running.1'

VersionIndependentProgID = s 'Running'

ForceRemove 'Programmable'

InprocServer32 = s "%MODULE%"

(

val ThreadingModel = s 'Apartment'

)

)

)

NoRemove AppID

(

NoRemove (E7531918-289D-11D2-869F-080009DC2552) = s 'Running Class'

(

val DllSurrogate = s ''

)

)

)

4/10

CRunning.h

FIG. 2C

```

// CRunning.h : Declaration of the CRunning
#ifndef _RUNNING_H_
#define _RUNNING_H_

#include "resource.h"      // main symbols

////////////////////////////////////
// CRunning
class ATL_NO_VTABLE CRunning :
    public CComObjectRootEx<CComSingleThreadModel>,
    public CComCoClass<CRunning, &CLSID_Running>,
    public IDispatchImpl<IRunning, &IID_IRunning, &LIBID_RUNNINGLib>,
    public IParseDisplayName
{
public:
    CRunning()
    {
        ATLTRACE(_T("CRunning() constructor called\n"));
    }
    virtual ~CRunning()
    {
        ATLTRACE(_T("CRunning() destructor called\n"));
    }
DECLARE_REGISTRY_RESOURCEID(IDR_RUNNING)
BEGIN_COM_MAP(CRunning)
    COM_INTERFACE_ENTRY(IRunning)
    COM_INTERFACE_ENTRY(IDispatch)
    COM_INTERFACE_ENTRY(IParseDisplayName)
END_COM_MAP()
////////////////////////////////////
//IParseDisplayName method
    STDMETHODIMP ParseDisplayName(IBindCtx *pbc
                                   ,LPOLESTR pszDisplayName
                                   ,ULONG *pchEaten
                                   ,IMoniker **ppmkOut
                                   );

protected:
    const wchat_t* ProgID() { return L"Running"; }
    const wchar_t* VersionIndependantProgID() { return L"Running.1"; }
// IRunning
public:
    STDMETHOD(RegisterInstanceName) (long lCookie);
    STDMETHOD(UnregisterInstanceName) (BSTR bstrItemName, IUnknown * pUnk, long * lCookie);
};
#endif // _RUNNING_H_

```

FIG. 2D

```

// CRunning.cpp : Implementation of CRunning
#include "stdafx.h"
#include "Running.h"
#include "CRunning.h"

#define BAD_POINTER_RETURN(p) if(!p) return E_POINTER
#define BAD_POINTER_RETURN_OR_ZERO(p) if(!p) return E_POINTER; else *p = 0
#define SIZE_OF_STRING(p) !p ? 0 : ((wcslen(p) * sizeof(wchar_t)) + sizeof(wchar_t))

#define OLE_MAXNAME_SIZE 256
////////////////////////////////////
// CRunning
STDMETHODIMP CRunning::RegistrationInstanceName(BSTR bstrItemName, IUnknown * pUnk, long * lCookie)
(
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    // TODO: Add your implementation code here
    ATLTRACE(_T("CRunning::RegisterInstanceName called\n"));
    HRESULT hr = E_FAIL;
    LPRUNNINGOBJECTTABLE prot = NULL;
    hr = GetRunningObjectTable(0, &prot);
    if(SUCCEEDED(hr))
    (
        LPMONIKER ppmk = NULL;
        hr = CreateItemMoniker(NULL, bstrItemName, &ppmk);
        if(SUCCEEDED(hr))
        (
            hr = prot->Register(0
                                .pUnk
                                .ppmk
                                ,(unsigned long *)lCookie
                                );
            if(SUCCEEDED(hr))
            (
                TRACE(_T("CRunning::RegisterInstanceName register succeeded cookie is %x\n"), (unsigned long *)lCookie);
            )
            else
            (
                TRACE(_T("CRunning::RegisterInstanceName register failed %x\n"), hr);
            )
            ppmk->Release();
        )
        else
        (
            TRACE(_T("CRunning::RegisterInstanceName CreateItemMoniker failed %x\n"), hr);
        )
        prot->Release();
    )
    else
    (
        TRACE(_T("CRunning::RegisterInstanceName get ROT failed %x\n"), hr);
    )
    return hr;
)

STDMETHODIMP CRunning::UnregisterInstanceName (long lCookie)
(
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    // TODO: Add your implementation code here
    HRESULT hr = E_FAIL;

```

```

if(!Cookie)
(
    LPRUNNINGOBJECTTABLE prot = NULL;
    hr = GetRunningObjectTable(0,&prot);
    if(SUCCEEDED(hr))
    (
        hr = prot->Revoke((unsigned long) lCookie);
        if(SUCCEEDED(hr))
        (
            TRACE(_T("CRunning::UnregisterInstanceName worked for cookie %x \n"),(uns
igned long)lCookie);
        )
        else
        (
            ATLTRACE(_T("CRunning::UnregisterInstanceName Revoke failed\n"));
        )
        prot->Release();
    )
    else
    (
        ATLTRACE(_T("CRunning::UnregisterInstanceName GetROT failed\n"));
    )
)
return hr;
)-

STDMETHODIMP CRunning::ParseDisplayName(
    IBindCtx* pbc,
    LPOLESTR pwszDisplayName,
    ULONG* pchEaten,
    IMoniker** ppmkOut)
(
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    ATLTRACE(_T("CRunning::ParseDisplayName() with %S/n"),pwszDisplayName);
    BAD_POINTER_RETURN_OR_ZERO(ppmkOut);
    BAD_POINTER_RETURN_OR_ZERO(pchEaten);
    BAD_POINTER_RETURN(pbc);
    BAD_POINTER_RETURN(pwszDisplayName);
    BAD_POINTER_RETURN(pchEaten);
    ATLTRACE(_T("CRunning::ParseDisplayName() pointers OK!\n"));
    HRESULT hr =E_FAIL;

    // set to max for now
    // need to change to fit MkParseEx
    if(*pwszDisplayName == L'0')
        *pchEaten = wcslen(L"0Running");
    else
        *pchEaten = wcslen(L"Running");
    // as far as i have been able to find out
    // MkParse will pass the 0, wrong!!
    // oh no!!! MkParseEx doesn't pass the 0!!
    // we've got to fix this, so let's look for ":"
    wchar_t * pwszInstance = wcschr(pwszDisplayName,L':');
    // do we have an instance?
    if(pwszInstance)
    (
        ATLTRACE(_T("CRunning::ParseDisplayName() instance name %S/n"),pwszInstance);
        WCHAR szItemName(OLE_MAXNAME_SIZE);
        LPWSTR lpszDest = szItemName;
        LPWSTR lpszSrc = pwszInstance;
        int cEaten = 0;
    )
)

```

7/10

```

                                CRunning.cpp
// eat delimiter characters until next token
while (*lpszSrc != L'\0' && (*lpszSrc == L'\' || *lpszSrc == L'/' ||
    *lpszSrc == L':' || *lpszSrc == L'!' || *lpszSrc == L'('))
(
    ++lpszSrc;
    ++cEaten;
)
// parse next token in szItemName
while (*lpszSrc != L'\0' && *lpszSrc != L'\' && *lpszSrc != L'/' &&
    *lpszSrc != L':' && *lpszSrc != L'!' && *lpszSrc != L'(' &&
    cEaten < OLE_MAXNAME_SIZE-1)
(
    *lpszDest++ = *lpszSrc++;
    ++cEaten;
)
*pchEaten += cEaten;
*lpszDest = 0;
// find the running object
LPRUNNINGOBJECTTABLE prot = NULL;
LPENMONIKER          penum = NULL;
LPMONIKER            ppmk = NULL;
hr = CreateItemMoniker(NULL, szItemName, &ppmk);
ATLTRACE(_T("CRunning::ParseDisplayName() CreateItemMoniker %x\n"), hr);
if (SUCCEEDED(hr))
(
    // look in the running object table to find the gizmo
    // since we are a moniker provider we can't use
    // the bind context to get the ROT
    hr = GetRunningObjectTable(0, &prot);
    ATLTRACE(_T("CRunning::ParseDisplayName() GetRunningObjectTable %x\n"), hr);
    if (SUCCEEDED(hr))
    (
        hr = prot->EnumRunning(&penum);
        ATLTRACE(_T("CRunning::ParseDisplayName() EnumRunning %x\n"), hr);
        if (SUCCEEDED(hr))
        (
            IMoniker * ppmkTest = NULL;
            IMoniker * ppmkResult = NULL;
            IUnknown * pUnk = NULL;
            BOOL bFound = FALSE;
            while((penum->Next(1, &ppmkTest, NULL) == S_OK) && (!bFound))
            (
                hr = ppmk->IsEqual(ppmkTest);
                if (hr == S_OK) // not SUCCEEDED!!
                (
                    TRACE(_T("CRunning::ParseDisplayName() we found it\n"));
                    bFound = TRUE;
                    hr = prot->GetObject(ppmkTest, &pUnk);
                    if (hr == S_OK) // not SUCCEEDED!!
                    (
                        TRACE(_T("CRunning::ParseDisplayName() we got it\n"));
                        hr = CreatePointerMoniker(pUnk, &ppmkResult);
                        if (SUCCEEDED(hr))
                        (
                            TRACE(_T("CRunning::ParseDisplayName() created pointer moniker\n"));
                            IParseDisplayName * pParse = NULL;
                            IMonikerpItem Moniker = NULL;
                            ULONG ucEaten

```

FIG. 2F

FIG. 26

8/10

CRunning.cpp

s correct
s to with it

```
(IID_IParseDisplayName, (void **)&Parse);
```

```
eDisplayName(pbc, lpszSrc, &ucEaten, &pItemMoniker);
```

```
+= ucEaten;
```

```
Result->ComposeWith(pItemMoniker, FALSE, ppmkOut);
```

```
DED(hr))
```

```
RACE(_T("CRunning::ParseDisplayName() It worked!!!\n"));
```

```
/ we can release the constituent elements
```

```
/ of the composite
```

```
pmkResult->Release();
```

```
succeed or fail we can release the
```

```
oniker
```

```
ker->Release();
```

```
;
```

```
// we'll give him the part that i
```

```
// and he can do whatever he want
```

```
*ppmkOut = ppmkResult;
```

```
// is there any string to parse?
```

```
if(*lpszSrc != L'\0')
```

```
(
```

```
hr = pUnk->QueryInterface
```

```
if(SUCCEEDED(hr))
```

```
(
```

```
hr = t pParse->Pars
```

```
if(SUCCEEDED(hr))
```

```
(
```

```
*pchEaten
```

```
hr = ppmk
```

```
if(SUCCEE
```

```
(
```

```
T
```

```
/
```

```
/
```

```
p
```

```
)
```

```
// if we
```

```
// item m
```

```
pItemMomi
```

```
)
```

```
pParse->Release()
```

```
)
```

```
)
```

```
)
```

```
)
```

```
)
```

```
ppmkTest->Release();
```

```
)
```

```
if(!bFound)
```

```
(
```

```
hr = E_FAIL;
```

```
)
```

```
penum->Release();
```

```
)
```

```
prot->Release();
```

```
)
```

```
ppmk->Release();
```

```
)
```

```
)
```

```
return hr;
```

```
)
```

FIG. 3

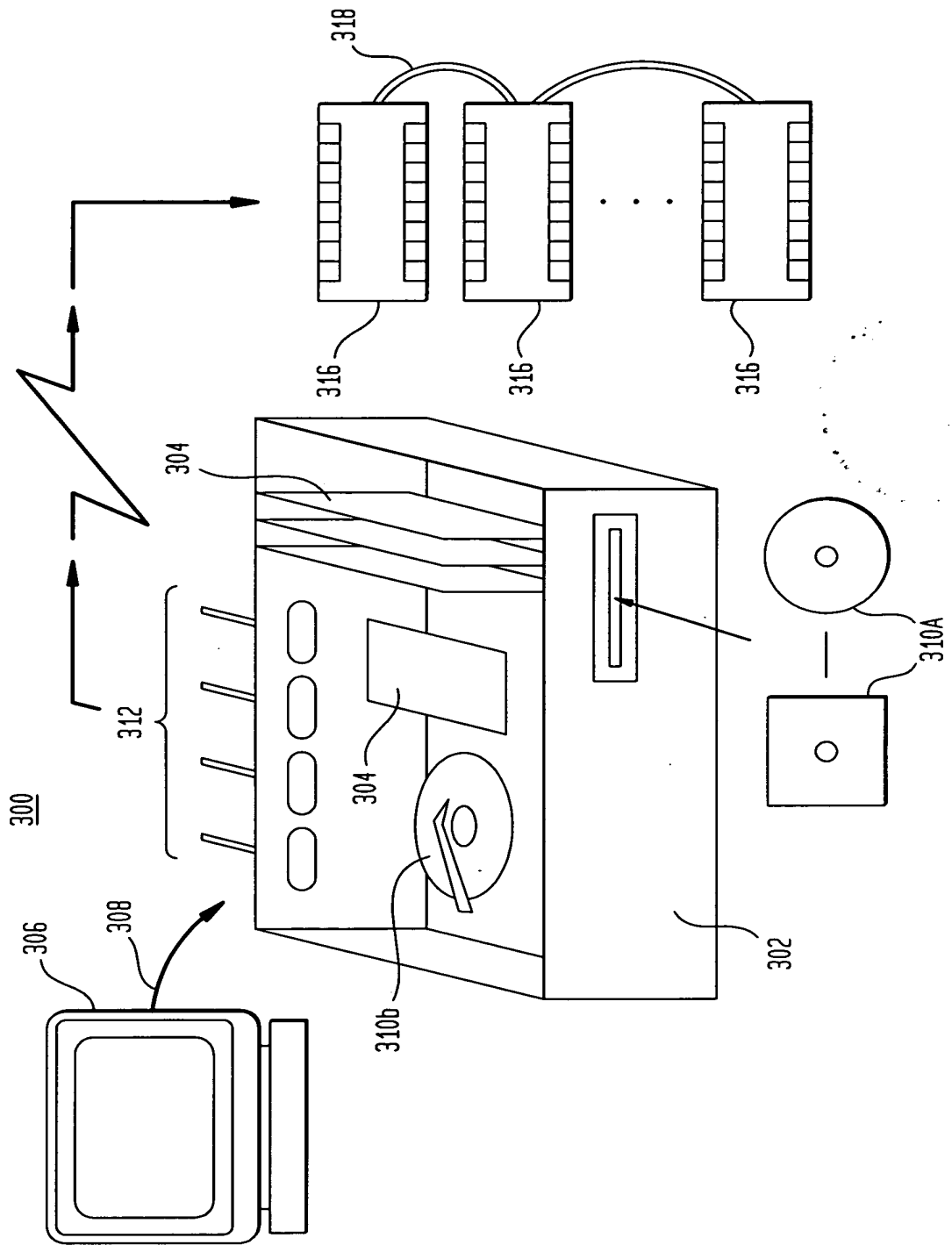


FIG. 4

